AZALEA

App participación ciudadana - Informe técnico

Control

April 13, 2024

Índice

1. Tarea

1.1. Objetivo perseguido

Existe una necesidad imperante de abordar los desafíos y motores socio-psicológicos, socio-técnicos e institucionales que impactan la participación efectiva del público en proyectos de infraestructura energética y urbana. Nuestra meta es explorar estrategias para superar las barreras existentes y lograr una participación significativa dentro de la comunidad universitaria.

Algunos problemas frecuentes en proyectos de participación ciudadana incluyen la falta de conexión entre el proyecto y las necesidades/intereses ciudadanos, lo que resulta en baja participación. Además, si los resultados no se traducen en acciones concretas o las expectativas no se cumplen, los ciudadanos pueden volverse desilusionados y menos propensos a participar en el futuro. La falta de incentivos tangibles o reconocimiento también puede disminuir el interés ciudadano.

En este contexto, surge la necesidad de proponer la creación de una aplicación web de participación ciudadana "Azalea - Participa". Esta iniciativa busca fortalecer la relación entre la comunidad y el tejido urbano, construyendo confianza y apoyo. La aplicación generará un mapa interactivo que recogerá no solo rutas habituales, áreas sin vegetación, zonas y espacios en mal estado, sino también las preocupaciones y sugerencias puntuales de la comunidad, a través de anotaciones digitales geolocalizadas que hará cada usuario desde su mapa. Tras un periodo de tiempo, las anotaciones individuales de cada usuario se recopilarán en el mismo plano, facilitando una percepción visual de los problemas y sugerencias más reportados, además de proporcionar una base de datos completa sobre la percepción de la comunidad universitaria, que usará nuestro equipo para desarrollar soluciones en el campus de Vera.

1.2. Descripción del trabajo realizado

Durante el desarrollo de la aplicación "Azalea - Participa", nos enfocamos en crear una plataforma interactiva que fomente la participación ciudadana en proyectos de infraestructura. Implementamos un mapa interactivo que permite a los usuarios abrir, comentar y respaldar incidencias en tiempo real. Hemos diseñado la interfaz de manera intuitiva para facilitar la geolocalización de problemas urbanos y recopilamos estas anotaciones para proporcionar una visión visual y detallada de las preocupaciones de la comunidad universitaria.

2. Trabajo realizado

2.1. Casos de uso

Los casos de uso son una herramienta fundamental en el desarrollo de software que permite definir, describir y comprender las interacciones entre los usuarios y un sistema o aplicación. Consisten en representaciones narrativas de cómo los usuarios interactúan con el sistema para lograr objetivos específicos. Estas representaciones proporcionan una visión clara y detallada de los requisitos funcionales de la aplicación, delineando las diversas acciones que los usuarios pueden llevar a cabo y cómo el sistema responde a estas acciones.

Los casos de uso delineados en este informe abarcan una amplia gama de interacciones entre los usuarios y la aplicación, desde la creación y el seguimiento de incidencias hasta la gestión y resolución por parte de los moderadores. Cada caso de uso está diseñado para optimizar la experiencia del usuario, garantizando una navegación intuitiva y una interacción fluida con la plataforma.

Se han realizado tres casos de uso, dividiéndolos en tipos de usuario.

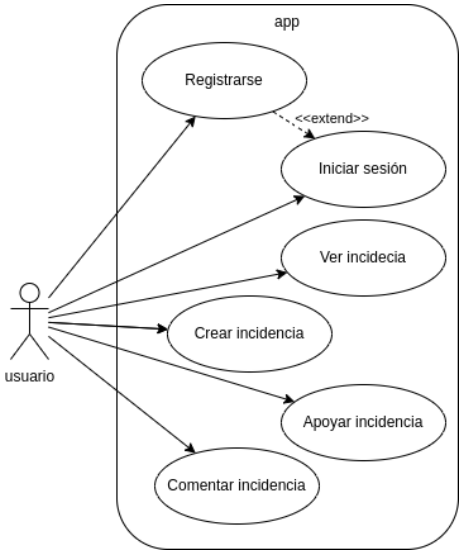

Figura 1: Casos de uso de usuario.

En la figura 1, se presentan los casos de uso destinados al usuario de la aplicación, delineando las diversas interacciones que este puede llevar a cabo. Estos casos de uso son esenciales para comprender cómo el usuario interactúa con el sistema y qué funcionalidades están disponibles para él.

El usuario de la aplicación cuenta con una serie de acciones a su disposición, las cuales se detallan a continuación:

- **Visualización de incidencias:** Este caso de uso permite al usuario explorar y visualizar las incidencias reportadas en el mapa. A través de esta funcionalidad, el usuario puede obtener información relevante sobre problemas o situaciones que requieren atención en ubicaciones específicas.
- **Registro e inicio de sesión:** En este caso de uso, el usuario tiene la opción de registrarse en la plataforma para acceder a funcionalidades adicionales o iniciar sesión si ya cuenta con una cuenta previamente creada. El registro y el inicio de sesión son pasos fundamentales para garantizar la seguridad y la personalización de la experiencia del usuario.
- **Creación de incidencias:** Mediante este caso de uso, el usuario puede reportar nuevas incidencias al sistema. Esto implica proporcionar detalles sobre la naturaleza del problema, adjuntar imágenes u otros archivos relevantes, y asignar una ubicación geográfica precisa para facilitar su posterior gestión y resolución.
- **Apoyo a incidencias:** Aquí, el usuario tiene la capacidad de expresar su apoyo o solidaridad hacia una incidencia específica. Esta función fomenta la participación y la colaboración entre los usuarios, permitiéndoles mostrar su interés y preocupación por determinadas situaciones.

- **Comentarios en incidencias:** Este caso de uso habilita al usuario para agregar comentarios adicionales a una incidencia existente. A través de esta funcionalidad, los usuarios pueden proporcionar información adicional, sugerir soluciones o simplemente expresar sus opiniones sobre un problema en particular, fomentando así la interacción y el intercambio de ideas dentro de la comunidad.

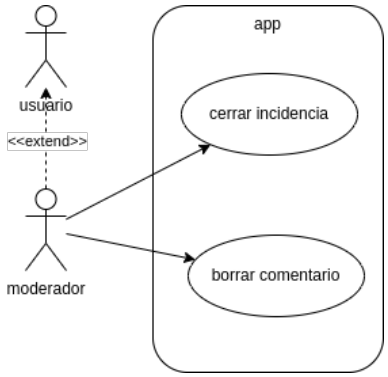

Figura 2: Casos de uso de moderador.

En la figura 2, se exhiben los casos de uso concebidos para los moderadores de la aplicación, detallando las funciones y responsabilidades específicas que tienen dentro del sistema. Estos casos de uso son detallan cómo los moderadores interactúan con la plataforma y cómo ejercen su rol de supervisión y gestión de contenidos.

Los casos de uso destinados a los moderadores incluyen:

- **Cierre de incidencias:** Este caso de uso permite al moderador finalizar o cerrar incidencias una vez que han sido debidamente atendidas o resueltas. Al cerrar una incidencia, el moderador indica que se han tomado las medidas necesarias para abordar el problema reportado, proporcionando así un seguimiento completo del ciclo de vida de la incidencia. También puede cerrar incidencias debido a que han sido indebidamente reportadas.
- **Borrado de comentarios:** En este caso de uso, el moderador tiene la capacidad de eliminar comentarios inapropiados, ofensivos o irrelevantes que hayan sido publicados en las incidencias. Esta función garantiza que el contenido dentro de la plataforma se mantenga adecuado y respetuoso, promoviendo un entorno de colaboración y participación positiva entre los usuarios.

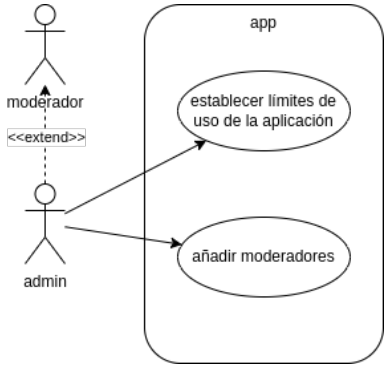

Figura 3: Casos de uso de administrador.

En la figura 3, se presentan los casos de uso diseñados específicamente para los administradores de la aplicación. Estos usuarios poseen privilegios adicionales que les permiten gestionar aspectos críticos del sistema y asegurar su correcto funcionamiento.

Los casos de uso para los administradores incluyen:

- **Establecimiento de límites geográficos:** Este caso de uso otorga al administrador la capacidad de definir y modificar los límites geográficos dentro de los cuales la aplicación estará disponible para su uso. Esto puede

incluir la delimitación de áreas específicas de servicio o la restricción de la aplicación a ciertos territorios, adaptando así la aplicación a las necesidades y requisitos particulares de cada contexto.

- **Adición de usuarios moderadores:** En este caso de uso, el administrador puede agregar nuevos usuarios al rol de moderador, otorgándoles así permisos y responsabilidades adicionales dentro del sistema. La capacidad de añadir moderadores es crucial para distribuir la carga de trabajo y garantizar una supervisión efectiva de las incidencias y los comentarios en la plataforma.

2.2. Desarrollo

Para este proyecto, desde control, se contaba con unos prototipos ya realizados en formato PDF (figura 4). Dados estos prototipos, se ha utilizado React para implementar la interfaz de la aplicación en formato web. Aunque ya se tenían prototipos, se han tenido que desarrollar otras vistas que no estaban prototipadas, como podrían ser la vista del login, o la vista para añadir una incidencia (figura 5). Desde el primer momento se ha tratado que dicha web sea usable tanto en dispositivos móbiles cómo en ordenadores. Para la parte del backend, se ha utilizado python junto con flask.

La aplicación ha sido estructurada de manera que el frontend y el backend funcionen como servicios separados. Esto significa que cada uno puede ser ejecutado en máquinas o entornos en la nube diferentes. En lugar de tener ambos componentes acoplados de manera inseparable, la arquitectura de la aplicación permite una mayor flexibilidad al permitir que el frontend y el backend operen de manera independiente.

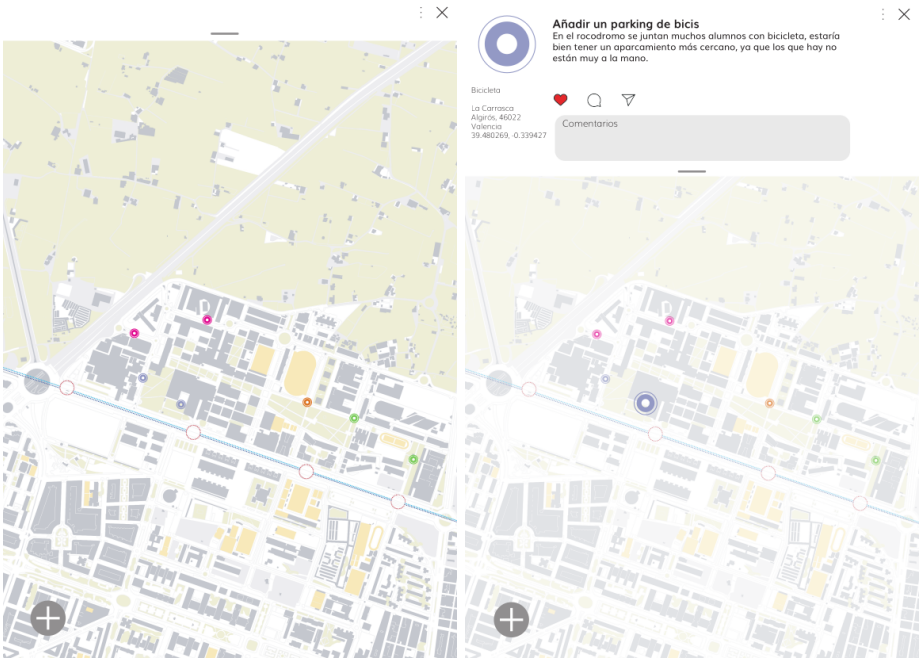

Figura 4: Prototipos previos al desarrollo.

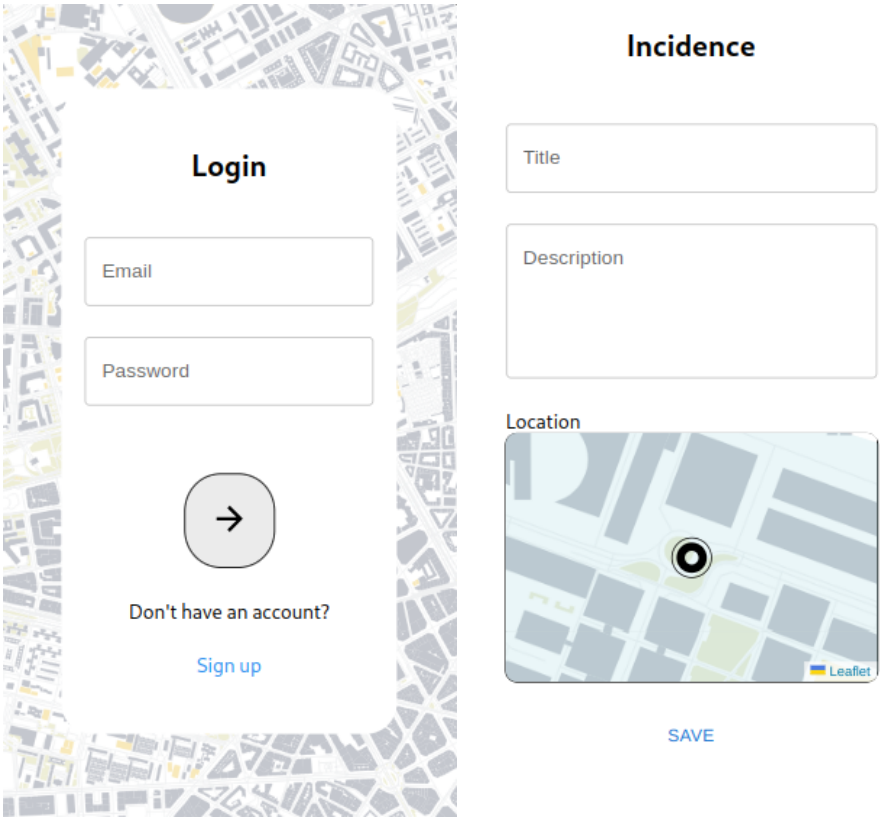

Figura 5: Prototipos creados para el desarrollo.

2.2.1. Obtención de mapas para la aplicación

El primer paso en el desarrollo de la aplicación fue la obtención de un mapa personalizado. Dado que los prototipos incluían mapas personalizados, se buscó una herramienta para la visualización de mapas en el frontend que permitiera la integración de mapas personalizados. Esta herramienta seleccionada fue Leaflet. Leaflet permite la incorporación de una URL personalizada para obtener los "Tiles"(teselas) que componen el mapa visible en pantalla.

En el contexto de Leaflet, los "Tiles"son imágenes cuadradas pequeñas que forman el mosaico visible del mapa. Estas imágenes son descargadas dinámicamente a medida que el usuario navega por el mapa. La utilización de "Tiles"facilita la visualización eficiente de mapas personalizados, ya que solo se descargan las partes necesarias, optimizando así la carga y la velocidad de respuesta de la aplicación.

La obtención, tanto de los mapas como de los tiles, se realizó utilizando la herramienta QGIS. QGIS es un sistema de información geográfica (SIG) de código abierto que ofrece funcionalidades robustas para la visualización, análisis y procesamiento de datos geoespaciales.

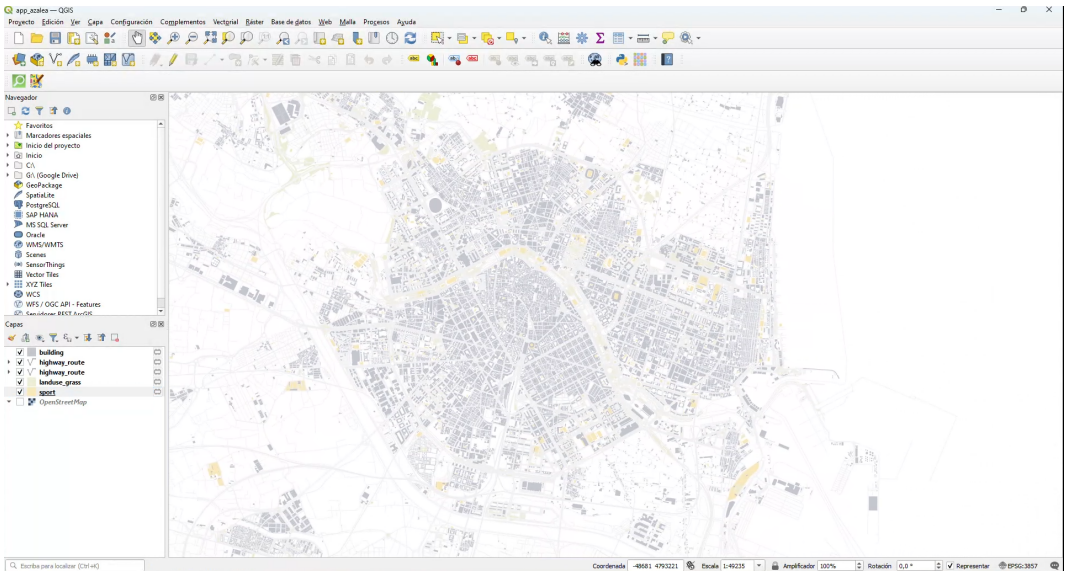

Figura 6: QGIS

Con QGIS y el plugin QuickOSM se obtuvieron las capas necesarias para la aplicación. Posteriormente, se editaron los colores de cada una de las capas, tal y como se muestra en la figura 6. Finalmente, se utilizó el plugin QTiles para la renderización de los tiles para la aplicación.

2.2.2. Desarrollo del frontend

Para el desarrollo del frontend, se han considerado las distintas vistas representadas en el diagrama de navegación del usuario, como se muestra en la figura 7.

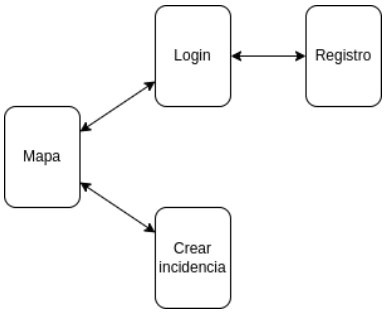

Figura 7: Diagrama de la navegación del usuario entre las vistas de la aplicación.

Las vistas contempladas son las siguientes:

- **Mapa:** Esta vista es la vista principal de la aplicación. Proporciona al usuario una representación visual de las incidencias reportadas en un mapa geográfico. Aquí, los usuarios pueden explorar las ubicaciones de las incidencias y acceder a detalles adicionales (como la descripción, comentarios o número de apoyos) haciendo click sobre ellas.
- **Login:** La vista de inicio de sesión permite a los usuarios autenticarse en la aplicación, proporcionando credenciales válidas, como nombre de usuario y contraseña, para acceder a funcionalidades exclusivas y personalizadas.
- **Registro:** En esta vista, los usuarios tienen la posibilidad de crear una nueva cuenta en la aplicación proporcionando la información requerida, como nombre, dirección de correo electrónico y contraseña. El registro es necesario para acceder a determinadas funcionalidades y para establecer una identidad única en la plataforma.

- **Crear incidencia:** Esta vista permite a los usuarios reportar nuevas incidencias en el sistema. Aquí, los usuarios pueden proporcionar detalles sobre el problema observado y especificar la ubicación geográfica exacta de la incidencia mediante un mapa interactivo.

Además de estas vistas, existe una vista adicional de **configuración**, en la que los usuarios administradores pueden establecer los límites geográficos del uso de la aplicación y gestionar los usuarios moderadores.

2.2.3. Desarrollo del backend

El backend backend de la aplicación se ha desarrollado en Python utilizando el framework Flask. Para la base de datos, se ha utilizado sqlite, ya que es una opción ligera y fácil de usar, adecuada para aplicaciones de tamaño medio. Flask proporciona una estructura flexible para desarrollar aplicaciones web, permitiendo la creación de rutas, la gestión de solicitudes y respuestas HTTP, y la integración sencilla con bases de datos y otras herramientas.

El backend se encarga de gestionar la lógica de negocio de la aplicación, incluyendo la autenticación de usuarios, la gestión de incidencias, la interacción con la base de datos y la comunicación con el frontend a través de API RESTful.

La gestión de incidencias se realiza a través de rutas RESTful que permiten realizar operaciones CRUD (Crear, Leer, Actualizar, Eliminar) sobre las incidencias almacenadas en la base de datos. Esto incluye la creación de nuevas incidencias, la recuperación de incidencias existentes, la actualización de la información de incidencias y el borrado de incidencias.

La comunicación entre el frontend y el backend se realiza mediante peticiones HTTP, utilizando el protocolo RESTful para definir las operaciones y los datos intercambiados entre ambos componentes. Esto permite una comunicación eficiente y escalable entre el frontend y el backend, garantizando una experiencia de usuario fluida y una gestión eficaz de los datos.

El backend se ha estructurado en los siguientes directorios:

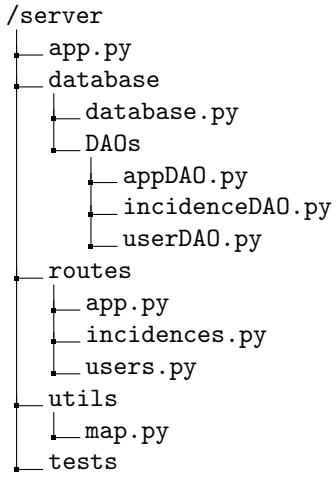

Figura 8: Estructura de directorios del backend

- **/server**: Directorio raíz del backend.
- **app.py**: Archivo principal que inicializa la aplicación Flask y configura las rutas y funcionalidades del servidor.
- **database**: Directorio que contiene los archivos relacionados con la base de datos.
- **database.py**: Archivo que define la configuración y la conexión con la base de datos SQLite.
- **DAOs**: Directorio que contiene los objetos de acceso a datos (DAO, por sus siglas en inglés) para interactuar con la base de datos.
  - **appDAO.py**: DAO para la gestión de la aplicación.
  - **incidenceDAO.py**: DAO para la gestión de incidencias.
  - **userDAO.py**: DAO para la gestión de usuarios.

- **routes**: Directorio que contiene los archivos de rutas, que manejan las solicitudes HTTP y definen las respuestas del servidor.
  - **app.py**: Archivo de rutas relacionadas con la aplicación en general.
  - **incidences.py**: Archivo de rutas relacionadas con las incidencias.
  - **users.py**: Archivo de rutas relacionadas con los usuarios.
- **utils**: Directorio que contiene utilidades o funciones auxiliares utilizadas en el backend.
  - **map.py**: Archivo que contiene funciones relacionadas con la manipulación de mapas o datos geoespaciales.
- **tests**: Directorio que contiene archivos de pruebas para realizar pruebas unitarias o de integración en el backend.

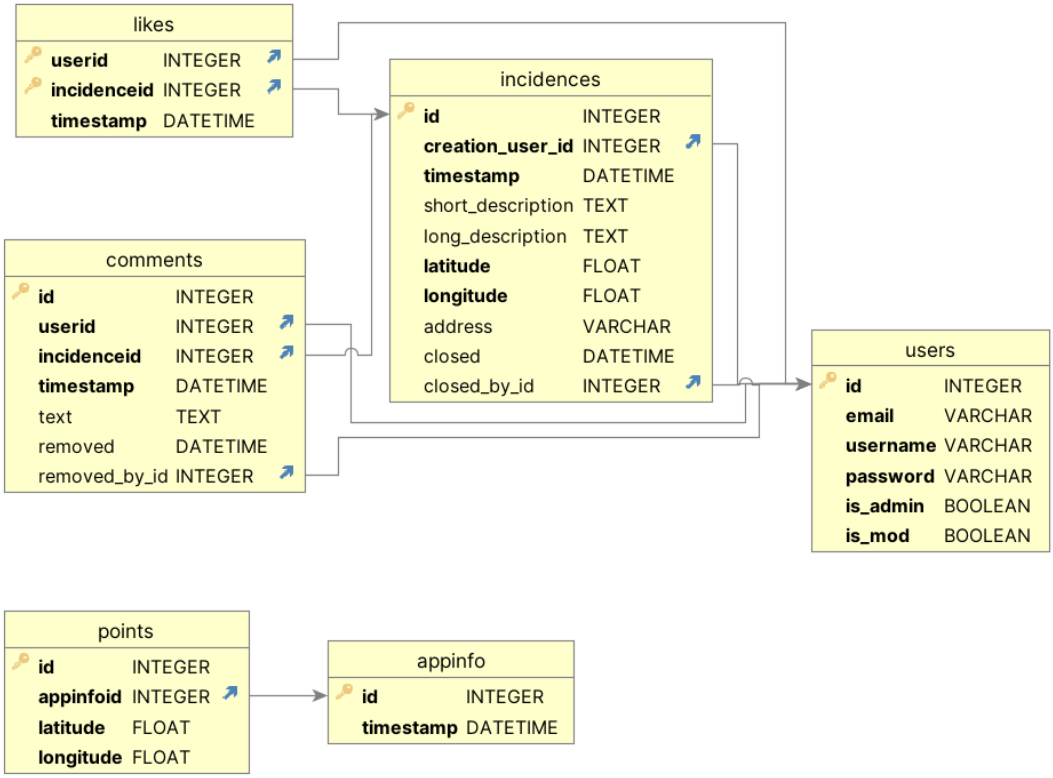

Figura 9: Diagrama de la base de datos.

En la figura 9 se puede apreciar el diagrama de la base de datos de la aplcación. En el diagrama se pueden apreciar las tablas principales de la base de datos y sus relaciones. A continuación, se describen brevemente estas entidades:

- **incidences:** Almacena información sobre las incidencias reportadas por los usuarios.
- **users:** Contiene información de los usuarios registrados en la aplicación.
- **points:** Se utiliza para almacenar información sobre los límites del mapa de la aplicación.
- **comments:** Guarda los comentarios realizados por los usuarios en las incidencias.
- **likes:** Registra los "me gusta" o apoyos de los usuarios a una incidencia específica.

3. Conclusiones

3.1. Logros

Durante el desarrollo del proyecto, se lograron varios hitos importantes:

- Se completó con éxito la implementación tanto del frontend como del backend de la aplicación, cumpliendo con los requisitos funcionales y no funcionales establecidos.
- Se diseñó una interfaz de usuario intuitiva y atractiva que facilita la participación de los usuarios y mejora la experiencia de usuario.
- Se estableció una conexión efectiva entre el frontend y el backend a través de una API RESTful, permitiendo una comunicación fluida y eficiente entre ambos componentes.
- Se implementaron pruebas unitarias y de integración en el backend para garantizar la calidad del código y la estabilidad del sistema.
- Se logró una estructura organizada y modular en el backend, facilitando el mantenimiento y la escalabilidad del sistema a largo plazo.

3.2. Trabajo Futuro

A pesar de los logros alcanzados, existen áreas que podrían mejorarse o explorarse en el futuro:

- Ampliar la funcionalidad de la aplicación para incluir características adicionales que mejoren la experiencia del usuario, como la integración con redes sociales, sistemas de notificación en tiempo real, etc.
- Explorar opciones para mejorar la seguridad de la aplicación, implementando medidas adicionales de autenticación, autorización y protección contra vulnerabilidades conocidas.
- Continuar mejorando y ampliando las pruebas automatizadas para garantizar la estabilidad y la calidad del código a medida que se realicen actualizaciones y adiciones al sistema.